An Adaptive Parametric Surface Mesh Generation Method Guided by Curvatures

Daniel M.B. de Siqueira[1], Markos O. Freitas[1], Joaquim B. Cavalcante-Neto[1], Creto A. Vidal[1], and Romildo J. da Silva[2]

[1] Department of Computing, Federal University of Ceará
{siqueira,markos,joaquimb,cvidal}@lia.ufc.br
[2] Department of Mathematics, Federal University of Ceará
rjsdusk@gmail.com

Summary. This work presents an adaptive mesh generation strategy for parametric surfaces. The proposed strategy is controlled by curvatures and the error measured between the analytical and discrete curvatures guides the adaptive process. The analytical curvature is a mathematical representation that models the domain, whereas the discrete curvature is an approximation of that curvature and depends directly on the used mesh. The proposed strategy presents the following aspects: it is able to refine and coarsen regions of the mesh; it considers the local error measures to ensure good global quality; it ensures good transition of the mesh and it deals with any type of parametric surfaces since it works in the parametric space.

Keywords: Curvatures, adaptivity, parametric surfaces, surface meshes.

1 Introduction

Surface meshing is a subject that has been studied for many years but it is still a topic of great interest in many areas, such as engineering and computer graphics. Two very important aspects to take into account for generating surface meshes are the runtime and quality. The runtime depends on the desired discretization, which is represented by the number of elements present in different regions of the model. The quality of the mesh, on the other hand, takes into account the quantity, distribution and shape of its elements [1, 2].

Adaptive meshing techniques are an interesting option to achieve quality for a mesh. In surfaces, the adaptive meshes have a higher concentration of elements in regions of greater curvature and a lower concentration of them in regions of lower curvature. The mesh adaptation improves the size, organization and arrangement of elements.

J. Sarrate & M. Staten (eds.), *Proceedings of the 22nd International Meshing Roundtable*,
DOI: 10.1007/978-3-319-02335-9_24, © Springer International Publishing Switzerland 2013

The techniques for adaptive meshes attempt to generate the mesh according to some criterion adopted, modifying the mesh where necessary. Among the most common changes there are: resizing, insertion and removal of elements. These changes are applied until a stopping criterion is reached. The criteria adopted for the modification must not be computationally expensive. In this work, an approach of low cost and good efficiency is adopted, which is the use of geometric criteria, as in [3, 4, 5, 6, 7]. Here, the curvature is considered as an important parameter in determining the density distribution of elements. The closer the discrete curvature measured in the mesh is to the analytical curvature, the mesh will be more faithful to the geometric characteristics of the surface.

Following this concept, this paper presents a strategy for generating adaptive meshes of parametric surfaces. It is a *posteriori* strategy, i.e., it starts from the initial geometry, and from this geometry, the strategy constructs a first approximation with a very coarse mesh. In each interaction, a new mesh, closer to the mathematical definition of the surface, is generated. The strategy presented seeks to ensure the following aspects: i) to be able to refine and coarsen regions of the mesh, ii) to ensure a smooth transition between regions of the mesh with higher and lower refinement, iii) to ensure compatibility between different regions, iv) to consider the contribution of the local error to measure the overall error in order to ensure a good quality for the final mesh.

This paper is organized into 5 sections. Section 2 presents some related work, and Section 3 presents an overview of the strategy. Section 4 shows some results achieved by generating meshes using the strategy. The results show its effectiveness and robustness in dealing with various levels of curvature. Finally, in Sect. 5, the conclusions about the strategy are presented.

2 Related Work

Curvatures in mesh generation have been extensively used in many areas, such as computer graphics and computer aided design. From a purely theoretical point of view, triangular meshes have no curvature, since all faces are flat and the curvature is not properly defined. But, by taking into account that a triangular mesh is an approximation of the solid surface, it is possible to estimate the curvatures by the geometric information in the mesh. The Gaussian curvature is one of the most known methods to estimate the curvature of a surface. Reference [7] presents some of the most widely used algorithms for calculating the Gaussian and mean curvatures. In that work, such algorithms are presented, analyzed and tested to determine what is the best alternative for the calculation of curvatures. Through comparative tests, the authors conclude that the best algorithms to calculate the Gaussian curvature are those that employ the Gauss-Bonnet scheme. Reference [6] analyses

the convergence of the scheme of Gauss-Bonnet, which was presented in [7]. The convergence of discretization scheme was considered in [4].

In [3] it is presented an error metric in order to generate several different mesh refinements to use in applications that involve levels of detail. This metric uses the method of discrete curvature as a criterion for simplification. The operators used follow the scheme of Gauss-Bonnet and can be used to calculate the curvatures for both the vertices located within the mesh and for those located on a border. Because of this feature, and for following the scheme of Gauss-Bonnet, such operators have been chosen for the calculation of discrete curvatures in the present work.

References [8, 9] present a technique for generating finite element meshes using the curvature as a measure to determine the size of the elements and guide the discretization. In [8] it is proposed a scheme for automatic generation of triangular meshes with arbitrary distribution of elements on curved surfaces and without the use of a parametric space. The second work [9] brings a similar approach, but proposes more control over the curvature.

Reference [10] presents a method for generating surface meshes using the parametric space and Delaunay triangulation. An initial set of points may be created in the parametric space, which can be random or entered manually. While this set of points is not mandatory, the starting point may be the parametric space square divided into two triangles, creating an initial very coarse mesh. The mesh error is measured and if this error is less than a certain precision ε and the number of triangles is less than a maximum, the triangle of greatest error is split at the point of greatest error according to the Delaunay criteria. The error of the mesh is the main point discussed in the article and techniques for measuring error are presented: the angle between the normal vectors of two adjacent triangles, an analytical error and a geometric one. None of these measures takes into account the curvature of the surface and is based on distances between a point on the surface and its parametric representation. Such approaches may have problems as this distance may not reflect the real geometry or indicate an error greater than the actual.

There are other works in the literature that address the generation of surface meshes, such as the ones presented in References [11] and [12], for example. Reference [11] generates parametric surface meshes using a combined advancing-front generalized Delaunay approach, whereas Reference [12] uses an advancing-front approach for the generation of the surface meshes using a Riemannian surface definition.

Reference [13] proposes an algorithm to obtain an optimal triangulated mesh using edge swapping in a sequential manner in order to reduce a cost function that measures the quality of the mesh. The cost function used by the authors is developed to take into account the curvature of the surface.

Reference [14] describes an algorithm to generate triangulated meshes of finite elements in arbitrary surfaces with high curvature. The mesh is generated in the parametric space and mapped to the 3D space. In this work, the

curvature taken into account is not calculated, since its values are given as an entry to the process. In another work by the same authors [15], an automatic algorithm for unstructured mesh regeneration on arbitrarily shaped three-dimensional surfaces is described. The algorithm works directly in Cartesian coordinates, as opposed to generating the mesh in parametric space.

With the awareness that normal vectors and curvatures are fundamental to the geometric modeling and computer-aided design, the work in [16] analyzed the numerical properties of parameterization-based computation of first and second-order differential quantities for continuous and discrete surfaces, and also compared a number of different methods for surface meshes in terms of accuracy, stability and efficiency.

As can be seen in [17], an upper bound of the discrepancy between the pointwise curvature of a curve and the discrete curvature of its polygonal approximation can be established. From an theoretical point of view, as the definition of the discrete Gaussian curvature at a vertex of a mesh is very analogous to the one for a polygonal line, the authors believe that is possible to derive an upper bound of the error between the discrete curvature and the Gaussian curvature of the surface.

The majority of these works and other works presented in the literature are able to refine regions of the mesh. Regarding the ability to coarsen regions of the mesh where there are more elements than necessary, which is sometimes also desired, there are much less works on the literature that are able to address this important aspect. They also do not ensure mesh compatibility between regions since they do not refine the curves independently from the domain. This can be very important in many applications and essential in applications where compatible meshes are required. Finally, although many of these works are in the parametric space, some of them do not allow the mesh to be generated for different mathematical description of the patches such as Hermite, Bézier and so on.

3 Adaptive Strategy

The adaptive strategy assumes that the surface to be meshed is composed of parametric patches, each of them bounded by parametric curves. In order to validate the strategy, in this work, Coons patches are used as the type of parametric surfaces to be meshed. Although this kind of patch is generally bounded by four parametric curves, the method can handle any kind of parametric patch, since what is relevant is the parametric coordinates. The strategy consists of two phases per iteration: first, given the geometric models of the patches and curves, as well as the initial mesh for the current iteration, the mesh quality measure based on error of curvature is computed; second, the mesh is modified by adding more elements in certain regions, and removing elements from other ones. The iterative process stops when a predefined global quality criterion is achieved. In phase 2, the curves are rediscretized

first and, only then, the patches are rediscretized. The strategy is discussed in details in the following.

3.1 Initial Model

The mathematical description of the patches together with the description of their boundary curves define the geometry, which is used in all steps of the adaptive strategy. The geometry description and an initial mesh compose the initial model that is used to start the adaptive process.

The geometric description is based on curves and patches: the descriptions of the curves come first, followed by the patches descriptions. In other words, two adjacent patches have at least one curve in common. This approach ensures the compatibility between two adjacent patches. The model depicted in Fig. 1 (a) describes a patch P_k and its curves C_i. This model is purely didactic and is used throughout this work to illustrate the adaptive strategy.

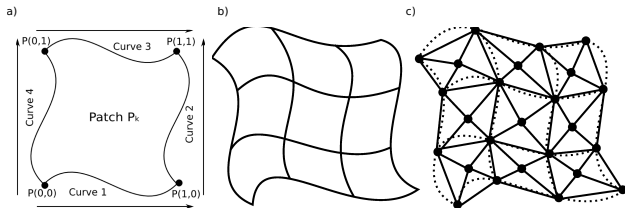

Fig. 1 Patch definition

The initial meshes of all patches compose the initial mesh of the model. It is a first approximation of the desired mesh and must be conformal. If no initial mesh is provided, it can be generated by the following way: if P_k is a patch and C_i represents its boundary curves, the generation of an initial P_k's mesh, M_k, starts with the discretization of C_i into segments. Thus, M_k is generated considering these segments, where the two end vertices of the segment form an edge, ensuring the compatibility of meshes at common borders of neighboring patches. The number of elements forming M_k depends on the discretization of P_k's boundary curves. In Fig. 1 (b), the discretization of the boundary curves in three segments each determines the discretization of the patch into nine regions, each one generating four triangular elements. The subdivisions of curves and patches are performed in parametric space and mapped into the three-dimensional space. It is important to mention, however, that this mesh is generated if no initial mesh is given, which is used instead. This makes the process completely generic for any given initial mesh.

3.2 Mesh Quality

The quality of every mesh generated during the adaptive process must be evaluated, according to an error criterion. Herein, a geometric criterion is adopted. It computes, at every mesh vertex, the error between the surface's curvatures evaluated both analytically and approximately. If the overall error distribution approaches zero in the average sense, the mesh quality is considered good, and the adaptation process stops. The local error indicates whether the mesh should be refined or coarsened at a particular region.

The analytic curvatures adopted to compute the error measures are the mean curvature $H = \frac{k_{min}+k_{max}}{2}$ and the Gaussian curvature $K = k_{min} \times k_{max}$, where k_{min} and k_{max} are the principal curvatures. The computation of H and K for a bi-parametric surface Q(u,v) can be found in [18] and elsewhere.

The discrete curvatures, unlike the computations of the analytical curvatures, which use the mathematical formulation of the patch, are evaluated based on the local information about the mesh. That is, the discrete curvatures are computed at a given mesh vertex and are based on the adjacent triangles' configurations. The discrete Gaussian and mean curvatures used in this work are computed with the discrete curvature operators presented in [3]. The discrete Gaussian curvature K for an inner vertex (Fig. 2a) is expressed as $K = \frac{2\pi - \sum \phi_i}{\frac{1}{3}A}$, where A is a sum of each triangles area, and ϕ_i denotes the angle at a vertex, both in three-dimensional space.

For a boundary vertex (Fig. 2b), the Gaussian curvature is defined by $K = \frac{\pi - \sum \phi_i}{\frac{1}{3}A}$, where A again represents the sum of each triangle's area and ϕ_i the angle at a vertex, both in three-dimensional space again.

The discrete mean curvature is given by $H = \frac{\Sigma m(e_i)}{\frac{1}{3}A}$, where e_i represents an edge of a vertex and $m(e_i)$ is a function of an edge e_i that returns the angle γ between two adjacent surface normal vectors, if the faces are convex (Fig. 2c), $-\gamma$, if the faces are concave, or 0, if the faces are planar.

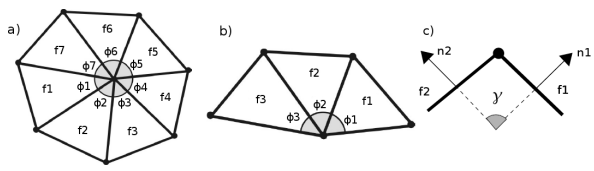

Fig. 2 (a) The Gaussian curvature of an inner vertex. (b) The Gaussian curvature of a boundary vertex. (c) The angle γ for the mean curvature.

3.3 Error Estimation

The adaptive process consists of curves and patches rediscretization. This is performed based on the error estimation of the discrete curvature relative

to the analytical curvature evaluated at the vertices. If the computed error indicates a high discrepancy between these curvatures, it suggests that the mesh must be locally refined or coarsened, based on a size parameter h.

Table 1 illustrates the possible scenarios and the determination of the new size parameter. k_a and k_d represent the analytic and discrete curvatures. The Gaussian curvature is the measure used to perform the error estimation, but when the Gaussian curvature is zero, the mean curvature is used instead.

In the first scenario, k_a is approximately equal to k_d. If these values are not close to zero ($k_a \nrightarrow 0$), the mesh captures well the local curvature of the surface and no refinement is necessary ($h_{new} = h_{old}$). However, if the mean curvatures k_a is close to zero ($ka \rightarrow 0$), the surface is locally planar and a coarsening of the mesh may be possible ($h_{new} = h_{old} * f$, $f > 1$).

Table 1 Error and estimation of element sizes

$k_a \approx k_d$	$k_a \rightarrow 0$	$h_{new} = h_{old} * f$	coarsening
$(\frac{k_a}{k_d} \rightarrow 1)$	$k_a \nrightarrow 0$	$h_{new} = h_{old}$	stop
$k_a \gg k_d$	$k_a \rightarrow 0$	$h_{new} = h_{old}/f$	refinement
	$k_a \nrightarrow 0$	$h_{new} = h_{old}/f$	refinement
$k_a \ll k_d$	$k_a \rightarrow 0$	$h_{new} = h_{old}/f$	refinement
	$k_a \nrightarrow 0$	$h_{new} = h_{old}/f$	refinement

In the other scenarios, the discrepancy between k_a and k_d is large. Then, when the mean curvature k_a is close to zero ($k_a \rightarrow 0$), the mesh is too coarse to capture the local flatness of the surface and should be refined. When k_a is not close to zero ($k_a \nrightarrow 0$), the mesh is again too coarse to capture the correct curvature and should be refined.

The factor f used for rediscretization is important to the rate of error decay. Which value of f to use is empirical at the moment and deserves more study in the future. In this work, a factor of 4 was used in the examples. This value experimentally showed good results. However, what is more important is that the error diminishes for many ranges of f.

Although the rediscretization is based on local errors from the curvatures, it is necessary a global measure to guide the whole iterative process. When a global quality measure is reached, the process stops. This measure implies the use of a global error estimate, η_{global}, which, in this work, is evaluated as

$$\eta_{global} = \frac{\sum_{j=1}^{N_v} \eta_j}{N_v}, \tag{1}$$

where N_v is the number of vertices of the whole mesh and η_j is the absolute value of the relative discrepancy between the analytical and the discrete curvatures at vertex j, computed as $\eta_j = \frac{|k_a - k_d|}{k_a}$.

The mesh has good quality when $\eta_{global} < \epsilon$. What value of ϵ is considered reasonable is a matter of judgment. Therefore, in this work, no suggestion is

made about what value of ϵ to use. It is simply shown that the errors decrease when the mesh is improved, indicating the efficiency of the iterative process.

3.4 Curve Adaptation

The first step of the presented strategy requires that the boundary curves of every patch be rediscretized prior to, and independent of, the rediscretization of the domain. It turns out that this is one of the main advantages of the strategy, since it leads to a more regular discretization of the patches' boundaries, consistent with the geometric characteristics of the surface near these curves. This is also important to ensure the mesh compatibility between two adjacent patches. The new discretization of the complete set of boundary curves defines the basis for the rediscretization of the domain, generating a new mesh for the model.

The boundary curves' rediscretization procedure is an one-dimensional version of the procedure based on a quaternary tree (quadtree) used to refine the domain [19]. It employs a recursive spatial enumeration technique, associated with a binary tree data structure, and adopts a criterion of curvature error to refine or coarsen a curve discretization, taking into account the curvature characteristics of its adjacent surfaces. Thus, at a given vertex of a curve's approximation polygon, discrete and analytic curvatures are computed. The discrete curvature considers all the triangles adjacent to the vertex that lie on the neighboring surfaces. The analytic curvature, on the other hand, considers the maximum of the curvatures computed for the surfaces sharing the curve. A curvature error measure guides the determination of new size of mesh triangles adjacent to that vertex.

The rediscretization of each curve is divided into three phases, as described in the following for a generic curve C_i.

Binary Tree Initialization

The first phase consists of initializing the binary tree that will guide the curve rediscretization. This tree is initialized with the minimum and maximum parametric coordinates (0, 1) of the non-discretized curve. This is done to allow a generic refinement procedure for any type of curve. The depth of the tree is initialized as zero, since the tree has only one level at this point.

Binary Tree Rediscretization

Each curve keeps the set of old adjacent vertices, which is set up when the initial mesh is generated and is updated after a new mesh is determined at each step. A curve also holds the set of new vertices that are generated during the current phase. Hence, these two sets have to co-exist until a new mesh is committed. These sets are sorted based on the parameterization of the curve.

The binary tree rediscretization of a curve (Algorithm 1) starts by traversing its sorted set of vertices from the old discretization. The process begins from the minimum and maximum parametric coordinates (0, 1). Every two consecutive vertices represent a segment of the curve. Each leaf of the tree stores the minimum and maximum parametric coordinates of the segment represented by that leaf. Both h_{par} and the vertex on the center of a curve's segment have to be considered in the parametric space because the curve is treated in this space.

Algorithm 1: Construction of the binary tree.

```
Initialize the binary tree with the root node empty ;
Assign min, max parametric coordinates (0,1) to the root;
Compute the length of the curve, L_curve;
for each segment defined by the old set of curve's vertices, S_k, k ← 1 to n_seg do
    Compute the segment's length in 3D space, L_seg_k;
    h_old ← L_seg_k;
    Compute the segment's center in 3D space, C_seg_k;
    Compute k_a at C_seg_k;
    Compute k_d as average of the k_d's at the endpoints,
    k_d = 1/2.(^{k-1}k_d +^{k1} K_d);
    Compute h_new according to the Table 1;
    // h_par ∈ [0,1]
    Compute h_par = h_new / L_curve;
    Determine the parameter u_k corresponding to C_seg_k;
    Determine in which cell of the tree u_k is located;
    while size of the cell > h_par do
        Subdivide the cell in two children;
        Increment the depth of the tree;
        Determine in which cell of the tree u_k is located;
```

After the curve segments' traversal, the resulting binary tree reflects the refinement according to the error in the curvatures computed at the vertices of the polygon approximation of curve C_i. Each leaf of the tree will generate a new segment of the curve's approximation polygonal line. These segments will be the sides of elements adjacent to the curve when the domain is rediscretized. Figure 3 illustrates an example of refinement of a curve C_i and its corresponding binary tree.

Update of the Discretization of the Curve Based on the Binary Tree

When the binary tree of curve C_i is finally redefined, the new discretization is included in the curve (Algorithm 2) and at the end of this process, the new

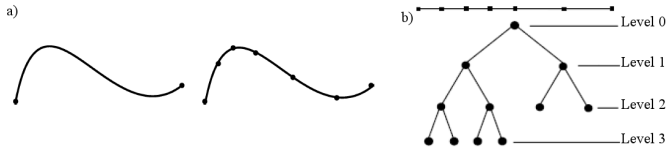

Fig. 3 (a) Refinement of a curve C_i and (b) its corresponding tree

Algorithm 2: Curve discretization.

```
Compute the 3D coordinates associated with u = 0;
Include these coordinates in the curve's structure;
for each leaf of the binary tree do
    Obtain the max parametric coordinate;
    Compute the corresponding 3D coordinates;
    Include these coordinates in the curve's structure;
```

curve discretization is obtained, in a sorted way. The new set of vertices is kept and the old set of vertices from the previous step is released.

3.5 Mesh Adaptation

After the discretization of the boundary curves, the next phase of the adaptive process generates a new mesh on the domain, using a procedure based on quadtree. However, in this work the quadtree is not responsible for the mesh generation in the whole domain. A strip close to the patch's boundary is left out to be discretized by an advancing-front technique. The procedure ensures that the mesh on the interior of a patch agrees with the discretization of its boundary curves. This is important because meshes on adjacent patches can be combined. It should be observed that, since the configuration of each curve will not change at this phase, the mesh generation of the patches can be implemented in parallel, although this was not done in this work.

The mesh generation by the quadtree procedure is based on [19], while the elements generated on the strip close to the boundary is generated by an advancing front technique, combined with a Delaunay criterion. The advancing front technique was modified in the present work to account for the quadtree information, in order to speed up the process by avoiding that an element is generated by a vertex far from its base edge. Figure 4 illustrates the boundary discretization for a given patch P_k. Since the procedure is based on a quadtree on a two-dimensional parametric space, it is necessary to map each vertex on the three-dimensional space to the parametric space of each patch. Moreover, at the end of the mesh generation, it is necessary to map the vertices of the elements generated in the parametric space back to the three-dimensional space.

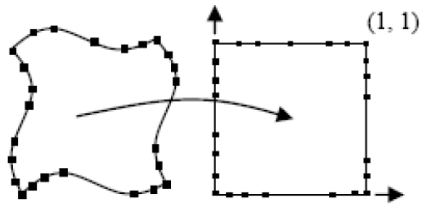

Fig. 4 Patch's curves discretization

The generation of each patch's new mesh occurs in three phases: the generation in the interior by a quadtree procedure; the generation in the transition zone by an advancing front approach and the smoothing of the generated mesh, as described in the following for a generic patch P_k.

Meshing in the Interior of the Domain by a Quadtree Procedure

Each patch P_k keeps the set of old adjacent elements, which is set up when the initial mesh is generated and is updated after a new mesh is determined at each step. A patch also holds the set of new elements that are generated during the current phase. Hence, these two sets have to co-exist until a new mesh is committed.

The quadtree procedure not only ensures that the interior density of the cells is sensitive to the boundary discretization but also guarantees a good transition between regions with different degrees of refinement. The procedure consists of the following steps: 1) Construction of the initial quadtree, 2) Adjustments due to errors of curvature for the elements, 3) Transformation of the tree into a restricted quadtree, 4) Elimination of cells close to the boundary, and 5) Generation of elements in the interior cells by patterns.

The initial quadtree is constructed taking into account the discretization of the patch's boundary curves obtained in the previous phase of the process (Fig. 4 shows the patch P_k in both 3D and parametric spaces). The construction process is performed in the parametric space according to the algorithm described in Algorithm 3. Figure 5 shows the initial quadtree associated with the boundary curve partition of Fig. 4.

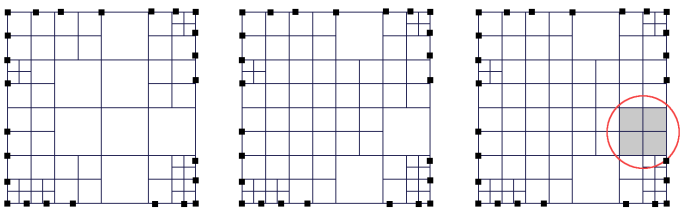

Fig. 5 Quadtrees: initial, adjusted by errors and restricted

The initial quadtree is then modified according to both the characteristics of the domain's mesh and the curvature errors at the mesh vertices. This is done by the procedure shown in Algorithm 4. In the evaluation of h_{old}, the area is computed in the 3D space. This helps to avoid distortions between the parametric and Cartesian spaces, which can occur since a square bounding box [(0,0) to (1,1)] is used. At the end of this step, the old set of elements is released. Figure 5 shows the adjusted quadtree after taking into account the error of curvatures in the domain's mesh.

Algorithm 3: Construction of the initial quadtree from boundary discretization.

Initialize the tree structure with the root node empty;
for *each boundary curve,* C_i*,* $i \leftarrow 1$ **to** 4 **do**
  Compute the length of the curve C_i *in 3D space,* L_i;
  for *each curve's segment,* S_j*,* $j \leftarrow 1$ **to** n_{seg} **do**
    Compute the size of S_j *in 3D,* L_{seg_j};
    Compute the segment's center in 3D space, C_{seg_j};
    Find the parametric coordinates of $C_{seg_j}, (u_j, v_j)$;
    Determine in which cell (u_j, v_j) *is located*;
    while *size of the cell* $> \frac{L_{seg_j}}{L_i}$ **do**
      Subdivide the cell into four children;
      Determine in which cell (u_j, v_j) *is located*;

Algorithm 4: Adjustments due to curvature.

$A_{total} \leftarrow 0$;
for *each element of the old set,* $E_k, k \leftarrow 1$ **to** n_{el} **do**
  $A_{total} = A_{total} + A_{E_k}$;
for *each element of the old set,* $E_k, k \leftarrow 1$ **to** n_{el} **do**
  Compute h_{new} *according to Algorithm 5*;
  Compute the center of E_k *in 3D space,* C_{E_k};
  Find parametric coordinates of C_{E_k}*,* (u_k, v_k);
  Determine in which cell (u_k, v_k) *is located*;
  while *size of the cell* $> h_{new}$ **do**
    Subdivide the cell into four children;
    Determine in which cell (u_k, v_k) *is located*;

Transforming the quadtree obtained in the previous step into a restricted quadtree is required because interior elements are created in interior cells of the quadtree by patterns. Moreover, only one level of difference between adjacent cells ensures a good mesh transition. Figure 5 illustrates the restricted quadtree. The marked cell, for instance, had to be divided.

Algorithm 5: Computing h_{new} for an element.

$h_{old_k} = \sqrt{\frac{A_{E_k}}{A_{total}}}$;
$h_{new} \leftarrow 0$;
for *each vertex of* $E_k, n_j, j \leftarrow 1$ **to** 3 **do**
  Compute k_a *and* k_d *at vertex* n_j;
  Compute h_{new_j} *as as in Table 1*;
  $h_{new} = h_{new} + h_{new_j}$;
$h_{new} = \frac{h_{new}}{3}$;

The next step is the triangulation of the cells. However, due to the boundary curves' discretization, not only the patterns cannot be applied but also distorted elements may be generated. In order to avoid this problem, a transition zone is created by the union of all cells which have contact with the boundary (Fig. 6). The cells that are not part of the transition zone will be meshed with the same patterns presented in [19] (Fig. 6). The cells that are in the transition zone are not used for meshing this region. Instead, an advancing front based on a Delaunay triangulation is used.

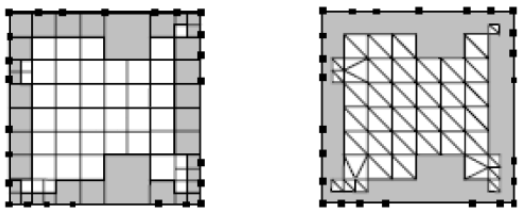

Fig. 6 The transition zone

The cell elimination process results in a quadtree composed of two types of cells: transition and interior cells. If the cell touches the boundary, it is a transition cell; otherwise, it is an interior cell. The leaves of the tree that are interior cells represent the sub-regions that will be meshed with the patterns. The use of patterns facilitates the generation of the interior mesh, accelerating the process. The subdivision made according to the curvature concentrates more elements, generated by patterns, in the most curved areas, mitigating the effect of distortion caused by the parameterization.

Meshing the Transition Zone by Advancing Front Procedure

The transition zone is meshed by an advancing front procedure that uses a Delaunay criterion. The advancing front was adapted to search only points in the adjacency of the chosen active edge, avoiding the unnecessary work of searching points occluded by the inner cells. This is the last phase of mesh

generation for patch P_k, and is performed as described in the Algorithm 6. In the example presented in this work, Fig. 7 illustrates the mesh after the advancing front in the parametric space, and this mesh is used to generate a corresponding mesh in the 3D space. Usually, the quality of the mesh is already good, but there are some distortions that can be fixed.

Algorithm 6: Advancing front.

```
Initialize the active edge list with boundary curves' segments;
while the active edge list is not empty do
    // for a new triangle's base
    choose an edge;
    for each chosen edge do
        Build a list of inner cells that are adjacent to the cells that contain the
        initial and final vertex of the chosen base edge;
        if there are no such inner cells then
            Build a list of cells that are in the 8-adjacency of the cells that
            contain the initial and final vertex of the chosen base edge;
    Find the best vertex from an adjacent cell list using a Delaunay criterion;
    Build the triangle;
    Update the active edge list;
```

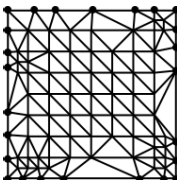

Fig. 7 Final mesh generated in parametric space

Finally, after the whole new mesh is completed in the parametric space, the 3D mesh is generated by mapping the parametric coordinates to the surface. Figure 8 shows the final mesh in 3D space.

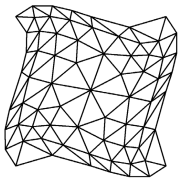

Fig. 8 Final mesh generated in 3D space, after the smoothing

4 Examples

In this section, a series of examples is shown. The patches are modeled as bi-parametric surfaces resulting from blending of the boundary curves. Since the strategy works directly with parametric surfaces, this approach gives independence to the technique to work with any kind of parametric surface. The stopping criterion used was the η_{global} (1). The ϵ used is 6%, but this value is user dependent. The zooming on the examples show how the elements are concentrated at regions where the curvatures are more accentuated.

The initial meshes were chosen as very coarse meshes. This choice was made on purpose, since the goal is to show the effectiveness of the strategy.

4.1 Description of the Examples

The first example shows a Hermite surface for a saddle with high curvature and non-planar border. Figure 9 shows the steps needed for the desired mesh generation. In this case, the error drops from 96% to 2.9%.

The second example shows a Hermite surface with a lower corner (Fig. 10), whose error drops from 92.7% to 5.4%.

The tire example (Fig. 11) is modeled with two Hermite patches. The first coarse mesh, which is more likely an octahedron than a tire, has 8 triangles and an error of 71.7%. The final mesh has 2196 elements and an error of 2.9%.

The last example, the Utah teapot (Fig. 12), is modeled using 32 Bézier patches. The first coarse mesh has 128 elements and an error of 58.2% and the final mesh, generated in 3 steps, has an error of 2.8% with 25162 elements.

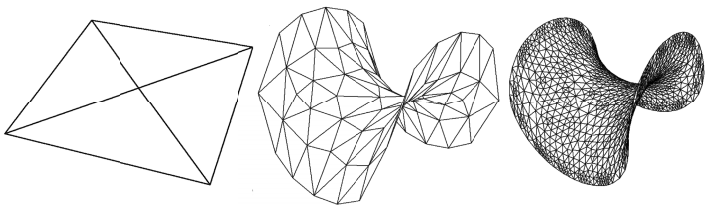

Fig. 9 The saddle's mesh: the initial mesh (5 vertices and 4 elements), the 1st step (65 vertices and 104 elements) and step 2 (1033 vertices and 1936 elements)

4.2 Analysis of the Examples

Table 2 shows, for the tire and the teapot models, that even with few steps, where the mesh is still coarse, the error drops drastically. This happens because the proposed strategy increases the number of elements in the correct regions, i.e., where there are great variations of curvatures. The process continues, but

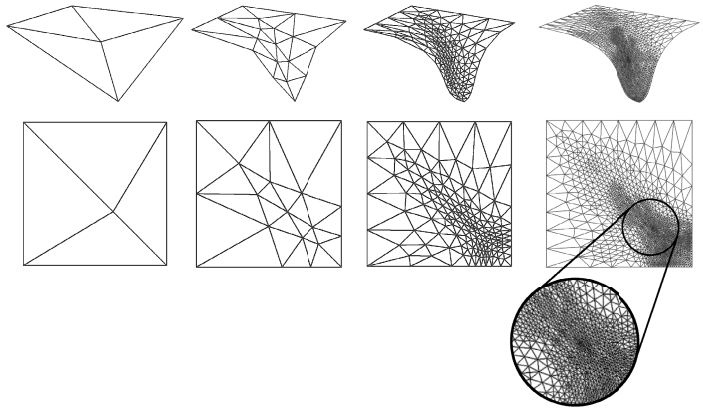

Fig. 10 The three steps needed for the lower corner surface

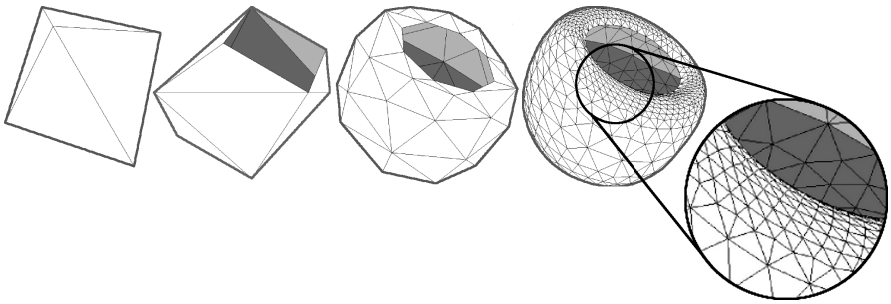

Fig. 11 The tire mesh generation

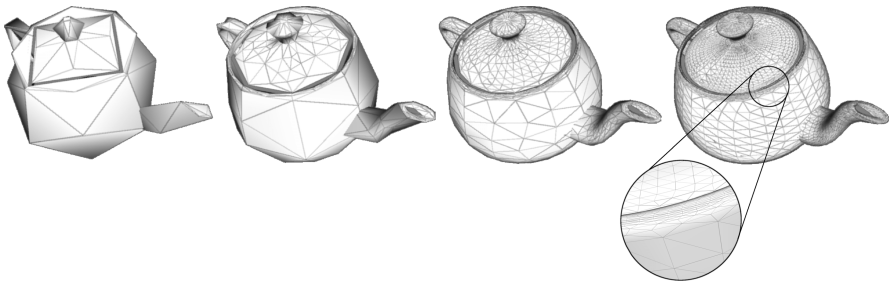

Fig. 12 The three steps for the teapot mesh generation

tends to stabilize as the mesh represents the geometry more accurately. Table 3 shows the growth of the number of elements and the decay of the errors for the initial and final meshes for most of the examples. It can be seen that the

Table 2 Number of Elements x Error

	Tire		Teapot	
Step	Elements	Error	Elements	Error
0	8	72%	128	58%
1	12	52%	836	30%
2	88	13%	4438	9%
3	440	6%	25162	3%
4	2196	3%	-	-

Table 3 Decay of the error in the examples

	Elements			Error		
Example	Initial mesh	Final mesh	Growth	Initial mesh	Final mesh	Decay
Saddle	4	1936	484%	96%	2.9%	93.1%
Lower corner	4	3470	867.5%	92.7%	5.4%	87.3%
Tire	8	2196	274.5%	71.7%	2.9%	68.8%
Teapot	128	25162	196.6%	58.2%	2.8%	55.4%

strategy presents a good error decay, making the mesh to get rapidly closer to the surface's actual geometry.

5 Conclusions

This work presented an adaptive mesh generation strategy for parametric surfaces. It is a curvature-controlled strategy which is able to refine and coarsen regions of the mesh and to ensure good mesh transition. It also ensures mesh compatibility between regions since it rediscretizes the curves independently of the domain. The strategy considers the contributions of local error measures to guarantee good global mesh quality and it works for any type of parametric surface. The examples shown in this work demonstrated that the adaptive strategy generates a good quality mesh even if the initial mesh is very coarse.

Some points are still of interest to this research, such as adjusting the method to consider anisotropic adaptivity and working on the implementation to handle different parameterizations, more general types of patches, and more than four parametric curves, in order to handle, for example, trimmed NURBS. Current efforts are being directed on a mathematical proof that the discrete mesh converges to the continuous surface patch.

References

1. Cheng, S.-W., Dey, T.K., Ramos, E.A., Ray, T.: Sampling and Meshing a Surface with Guaranteed Topology and Geometry. In: Proceedings of the Twentieth Annual Symposium on Computational Geometry, pp. 280–289 (2004)
2. Boissonnat, J.-D., Oudot, S.: Provably good sampling and meshing of surfaces. Graph. Models 67(5), 405–451 (2005)
3. Kim, S.J., Jeong, W.K., Kim, C.H.: LOD Generation with Discrete Curvature Error Metric. In: Proceedings of Korea Israel Bi-National Conference, pp. 97–104 (1999)
4. Meek, D., Walton, D.: On surface normal and Gaussian curvature approximations given data sampled from a smooth surface. Computer Aided Geometric Design 17, 521–543 (2000)
5. Meyer, M., Desbrun, M., Schröder, P., Barr, A.H.: Discrete Differential-Geometry Operators for Triangulated 2-Manifolds. Visual and Mathematics III, 35–57 (2002)
6. Xu, G.: Convergence Analysis of a Discretization Scheme for Gaussian Curvature over Triangular Surfaces. Computer Aided Geometric Design 23(2), 193–207 (2006)
7. Magid, E., Soldea, O., Rivlin, E.: A Comparison of Gaussian and Mean Curvature Estimation Methods on Triangular Meshes of Range Image Data. Computer Vision and Image Understanding 107(3), 139–159 (2007)
8. Lo, S.H., Lau, T.S.: Finite Element Mesh Generation over Analytical Curved Surfaces. Computers and Structures 59(2), 301–309 (1996)
9. Lo, S.H., Lau, T.S.: Mesh Generation over Curved Surfaces with Explicit Control on Discretization Error. Engineering Computations 15(3), 357–373 (1998)
10. Seibold, W., Wyvill, G.: Towards an Understanding of Surfaces through Polygonization. In: Computer Graphics International Conference, p. 416 (1998)
11. Borouchaki, H., Laug, P., George, P.L.: Parametric surface meshing using a combined advancing-front generalized Delaunay approach. International Journal for Numerical Methods in Engineering 49(1-2), 233–259 (2000)
12. Tristano, J.R., Owen, S.J., Canann, S.A.: Advancing front surface mesh generation in parametric space using a Riemannian surface definition. In: Proceedings of 7th International Meshing Roundtable, Sandia National Laboratories, pp. 429–445 (1998)
13. Dyn, N., Hormann, K., Kim, S.J., Levin, D.: Optimizing 3D triangulations using discrete curvature analysis. Mathematical Methods for Curves and Surfaces, pp. 135–146 (2000)
14. Miranda, A.C.O., Martha, L.F.: Mesh Generation on High-Curvature Surfaces Based on a Background Quadtree Structure. In: Proceedings of 11th International Meshing Roundtable, Sandia National Laboratories, pp. 333–342 (2002)
15. Miranda, A.C.O., Martha, L.F., Wawrzynek, P.A., Ingraffea, A.R.: Surface mesh regeneration considering curvatures. Engineering with Computers 25(2), 207–219 (2009)
16. Wang, D., Clark, B., Jiao, X.: An analysis and comparison of parameterization-based computation of differential quantities for discrete surfaces. Computer Aided Geometric Design 26(5), 510–527 (2009)

17. Borrelli, V., Orgereta, F.: Error Term in Pointwise Approximation of the Curvature Term of a Curve. Computer Aided Geometric Design 27(7), 538–550 (2010)
18. Rogers, D.F., Adams, J.A.: Mathematical elements for computer graphics, 2nd edn., pp. 420–421. McGraw-Hill Science/Engineering/Math (1990)
19. Wittchen, S., Baehmann, P., Shephard, M.: Robust geometrically based, automatic two-dimensional mesh generation. International Journal for Numerical Methods in Engineering 24, 1043–1078 (1987)